Latent Regularization in Generative Test Input Generation

Anonymous Author(s)

Abstract

This study examines how the impact of regularization of latent spaces through truncation affects the quality of generated test inputs for deep learning classifiers. We evaluate this effect using style-based GANs, a state-of-the-art generative approach, and assess quality along three dimensions: validity, diversity, and fault detection. We evaluate our approach on the boundary testing of deep learning image classifiers across three datasets — MNIST, Fashion-MNIST, and CIFAR-10. We compare two truncation strategies: latent code mixing with binary search optimization and random latent truncation for generative exploration. Our experiments show that the latent code–mixing approach achieves a higher fault detection rate than random truncation, while also improving both diversity and validity.

1 Introduction

Testing deep learning (DL) systems requires large, diverse, and valid inputs to expose misbehaviors [7, 16, 20, 24, 25, 31, 34]. As deep learning models and datasets grow in complexity, manually creating such inputs becomes infeasible due to the huge size of the input space. Test input generators can be used to sample from this space more effectively. Recent generative AI methods, such as style-based GANs, are promising as they produce high-quality synthetic data and enable structured manipulations in their latent spaces [6, 12, 14, 15]. Generating test inputs requires balancing two conflicting objectives: the inputs should be valid (plausible and human-recognizable) but also diverse enough to explore the DL system decision space and reveal faults.

Previous work has used GANs to generate test inputs with semantic variations (for example, weather changes in autonomous driving) [35], using mechanisms such as *truncation* to balance realism and diversity [3, 21]. Truncation works by moving latent codes toward their mean, producing more realistic but less diverse images. Although truncation is part of most modern GANs [13, 15, 29], its effect on generating test inputs for DL systems has not been thoroughly analyzed. Existing DL testing work usually relies on pixel or feature perturbations [7, 25], or explores semantic changes without considering truncation as a controllable factor [31, 35]. Recent search-based approaches [5, 11, 26, 32, 37, 38] explore other forms of variability, but not this specific trade-off.

This work explores how truncation, a common latent-space regularization method, influences the quality of test inputs produced by class-conditional StyleGAN2 models. These models form the foundation of recent generative testing frameworks, such as `Mimicry`, which we adopt as a reference in this study [32]. We first perform a screening step on the generated input seeds to filter out invalid samples, using a combination of classifier confidence, output margin, SSIM, and L2 threshold [9, 18, 27, 28]. We further evaluate an adaptive truncation strategy, in which we progressively decrease the truncation rate until valid candidate seeds are produced. These "salvaged" seeds typically lie close to the classifier's decision boundaries and are therefore more likely to trigger misclassifications. Using these seeds, we then generate failure-inducing inputs with Mimicry through a search-based test generation procedure.

We conduct experiments on three datasets, namely MNIST, Fashion-MNIST, and CIFAR-10, varying the truncation level to study its effect on multiple quality metrics: (i) the proportion of samples passing automatic screening, (ii) the fraction of human-validated valid samples, (iii) the diversity among validated samples, and (iv) the fault detection rate among validated inputs.

Our experiments demonstrate the advantage of the adaptive seed selection strategy. In the MNIST benchmark targeting 25 frontier pairs, 15 samples were confirmed as valid and fault-revealing, with 12 of them originating from salvaged seeds. Overall, our results indicate that adaptive truncation improves the acceptance of generated test inputs while maintaining high diversity and increasing fault detection effectiveness.

Our paper makes the following contributions:

Systematic study of truncation for test generation. We conduct the first structured investigation of how latent-space regularization through truncation affects the quality of generated test inputs for deep learning classifiers.

Adaptive truncation strategy for seed selection. We propose an adaptive truncation method that progressively lowers the truncation parameter ψ to salvage seeds near decision boundaries, improving fault discovery while preserving diversity.

Experimental evaluation. We evaluate both random and search-based truncation strategies on three datasets, MNIST, Fashion-MNIST, and CIFAR-10, measuring validity, diversity, and fault detection under different truncation levels. Our experiments show that adaptive truncation improves the acceptance rate and fault detection capability of generated test inputs, offering practical guidance for designing generative testing frameworks.

2 Background

2.1 DL Testing Objectives

Testing of DL systems generally serves two goals: *robustness testing*, which stresses a trained model using semantically preserved perturbations, and *generalization testing*, which seeks novel yet *valid* inputs that traverse or expose decision boundaries [7, 20, 25, 31, 34].

Boundary testing is a form of generalization testing that focuses on regions of near-ambiguity, where class confidences are similar. It can be performed in an *untargeted* way (searching for any label flip) or *targeted* (moving from a source class toward a specific rival suggested by the model's confidence distribution). Targeted exploration is usually more sample-efficient in high-dimensional spaces, as it constrains the search to a source–rival corridor on the decision surface [2, 4, 22, 32].

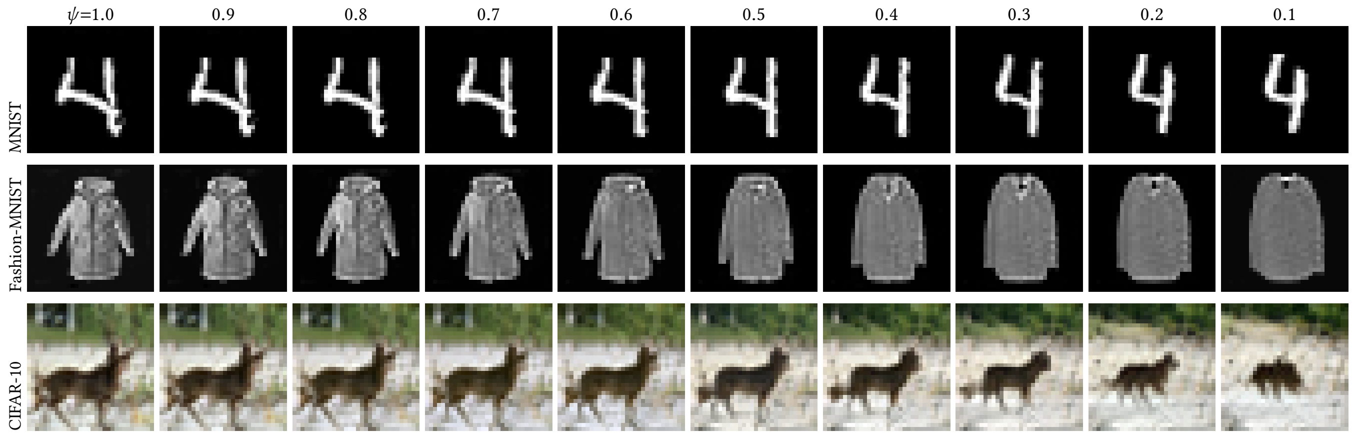


Figure 1: Truncation examples (class 4 from each dataset) on coarse layers. A single latent seed is rendered at $\psi = 1.0$ and then with decreasing ψ (z-latent frozen). Lower ψ shifts intermediate latents toward $\bar{w}$, increasing fidelity and reducing diversity.

2.2 Test Input Generation Families

Input-based methods operate directly on the input space, perturbing existing test samples to reveal faults. Examples include gradient-based techniques such as DeepXplore [25] and DLFuzz [7], which apply controlled perturbations to increase neuron coverage or induce misclassifications, as well as random or search-based image transformations [31]. While these methods are simple and model-agnostic, they only target robustness testing at the model level, limiting their ability to expose system-level faults.

Model-based methods, on the other hand, define explicit domain models that are explored to populate challenging regions with valid inputs [26, 37, 38]. They offer interpretability and control but depend on the abstraction representativeness of the underlying model. For example, DeepJanus and DeepHyperion leverage domain-level representations to guide search, enabling more structured exploration of boundaries of behaviours.

Finally, latent-space methods use a generative model to define a continuous manifold of realistic inputs, and test generation explores this space to produce diverse and plausible samples [11, 32]. Recent GenAI-based frameworks, such as Mimicry [32], RBT4DNN [23], CIT4DNN [5], and GIFTBench [21], use different forms of latent-space exploration, including interpolation, noise injection, and guided search, to generate semantically valid and behavior-revealing test inputs.

In this work, we focus on the third family and isolate the role of a single, widely used latent-space parameter, i.e., *truncation*, as a controllable factor influencing the validity, diversity, and fault revelation of generated test inputs. To analyze the effect of this hyperparameter, we build upon Mimicry, a recent generative testing framework that serves as the reference test generator in this study. We first summarize the main functioning of Mimicry to contextualize our experimental setup and clarify how truncation interacts with its boundary-search process.

Mimicry builds upon a class-conditional StyleGAN2 model to disentangle the latent space into *style* and *content* components, making StyleGANs practical back-ends for latent-space test generation. The generator maps an initial latent vector $z \in \mathcal{Z}$, sampled from a Gaussian distribution, into an intermediate latent space $\mathcal{W}$ through a mapping network. This transformation organizes features by scale across layers, enabling precise control over the generated content. The layer-wise interface of $\mathcal{W}$ and its extended version $\mathcal{W}^+$ supports scale-specific modifications through *style mixing* [1, 10, 14, 30] or *feature mixing* [32]. Mimicry employs a search-based generation process guided by a fitness function to locate *boundary inputs*, samples that lie near the model's decision boundary, where class probabilities are balanced (e.g., softmax outputs close to 50–50 between two classes or evenly distributed across several classes).

3 Methodology

In generative test input generators such as Mimicry, the truncation parameter ψ can be used as a regularization control that moves each latent code toward the average latent $\bar{w}$, trading diversity for realism. The so-called *truncation trick* contracts latent codes w toward the running average $\bar{w}$ according to:

$$w' = \bar{w} + \psi(w - \bar{w}), \qquad \psi \in (0, 1].$$

Lower values of ψ typically increase visual realism and classifier confidence but reduce the diversity of generated samples [3, 15, 29]. This makes truncation a natural variable for studying how fidelity controls influence the generation of valid and fault-revealing test inputs. However, test input generation must also balance other competing goals: *validity*, i.e., producing class-plausible images, and *diversity*, i.e., covering enough of the input space to traverse decision boundaries and expose faults.

3.1 Approach

Truncation in StyleGAN can be applied uniformly across all layers in the latent code w or on a per-layer basis in $\mathcal{W}^+$ [15]. Uniform schedules simplify comparisons, while layer-wise schedules allow different truncation values per layer to preserve coarse semantic structure [1, 14, 30]. Lowering the truncation parameter ψ concentrates samples in high-density regions, typically increasing fidelity and classifier confidence while reducing diversity. An example is given in Fig. 1.

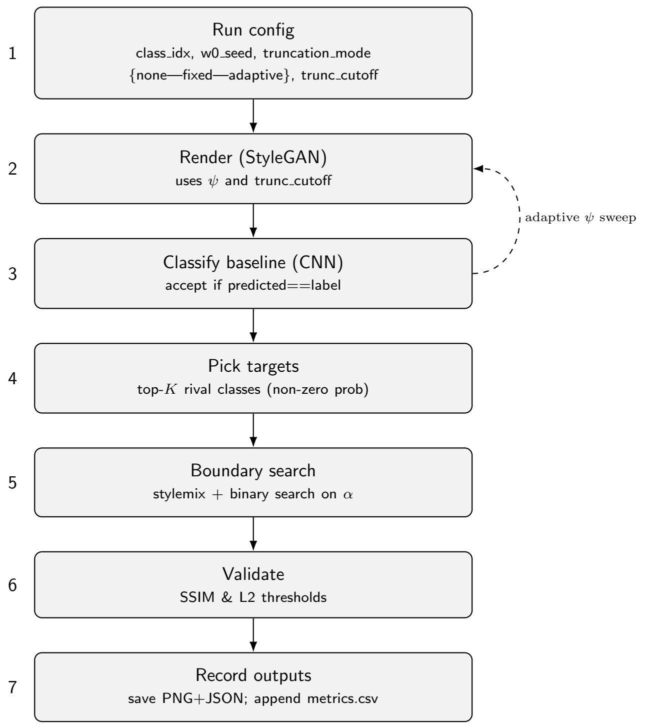

1
Run config
class_idx, w0_seed, truncation_mode
{none—fixed—adaptive}, trunc_cutoff
2
Render (StyleGAN)
uses ψ and trunc_cutoff
adaptive ψ sweep
3
Classify baseline (CNN)
accept if predicted==label
4
Pick targets
top-K rival classes (non-zero prob)
5
Boundary search
stylemix + binary search on α
6
Validate
SSIM & L2 thresholds
7
Record outputs
save PNG+JSON; append metrics.csv

Figure 2: Test input generation with truncation workflow.

We use two key notions throughout our methodology. The *minimal acceptable truncation* $\psi^{\star}$ is the smallest ψ at which a seed remains behavior-preserving for the DL classifier; this forms the basis of *seed salvage*. The *first-flip truncation* is the first ψ (descending from 1.0) at which the classifier's prediction changes.

Our test input generation workflow is illustrated in Fig. 2. Each run is configured with a class index, latent seed, truncation mode (*none, fixed, adaptive*), and truncation cutoff, specifying the feature layer up until truncation will be applied starting from the bottom, giving an option to truncate only lower layers and leave fine details unchanged Fig. 7. Images are rendered at the chosen ψ with the initial latent z frozen to isolate truncation effects [15]. The baseline check at $\psi = 1.0$ ensures that only seeds passing the fixed confidence and margin requirements are retained. For boundary testing, we select the top-K rival classes according to the DL classifier probabilities, whenever we have non-zero probabilities for classes other than source. Each candidate then undergoes boundary exploration using one of two techniques: truncation-assisted style mixing or truncation-only first-flip truncation. After the search, candidates are evaluated based on SSIM and L2 thresholds to filter out invalid frontier pairs early on.

In *truncation-assisted style mixing* (Algorithm 1), the source class latent $w_c^{(b)}$ is mixed with a rival latent $w_r^{(b)}$ over a specified set of layers L [14]. A binary search on the mixing weight $\lambda \in [0, 1]$ identifies a minimal-flip edit while respecting the SSIM and L2 threshold (1). In *truncation-only first-flip* (Algorithm 2), ψ is decreased from 1.0 along a predefined schedule until the classifier prediction flips.

Algorithm 1 Boundary Testing via Truncation-Assisted Style Mixing

```
Require: Generator $G$, classifier $h$; budgets $B$ ($\psi_T$); seeds $\mathcal{Z}$; rivals
    top-$K$; layers $L$; binary steps $T$; SSIM & L2 threshold $\tau$.
 1: for $z \in \mathcal{Z}$ do
 2:     $x_{1.0} \leftarrow G(z, c; 1.0)$
 3:     $R \leftarrow$ top-$K$ rivals by confidence on $x_{1.0}$
 4:     for $b \in B$ do ▷ truncation budgets
 5:         $x_b \leftarrow G(z, c; b)$
 6:         for $r \in R$ do
 7:             binary search $\lambda$; $x_\lambda \leftarrow \mathrm{StyleMix}(w_c^{(b)}, w_r^{(b)}, L, \lambda)$
 8:             if (SSIM & L2) $\leq \tau$ and $\arg\max h(x_\lambda) \neq c$ then
 9:                 record frontier $(x_b, x_\lambda, b, \lambda)$; break
10: Aggregate: frontiers with corresponding JSON files.
```

Algorithm 2 Truncation-Only First-Flip Search

```
Require: Generator $G$; classifier $h$; class $c$; schedule $\Psi$ =
    $\{1.0, \psi_2, \ldots, \psi_M\}$; seeds $\mathcal{Z}$.
 1: for $z \in \mathcal{Z}$ do
 2:     $x_{1.0} \leftarrow G(z, c; 1.0)$ ▷ base input
 3:     if $\arg\max h(x_{1.0}) \neq c$ then
 4:         then continue
 5:     for $\psi_T \in \Psi \setminus \{1.0\}$ (descending) do
 6:         $x_{\psi_T} \leftarrow G(z, c; \psi_T)$
 7:         if $\arg\max h(x_{\psi_T}) \neq c$ then
 8:             record $\psi^\star = \psi_T$, $(z, c, x_{\psi_T})$; break
 9: Aggregate: frontiers with corresponding JSON files.
```

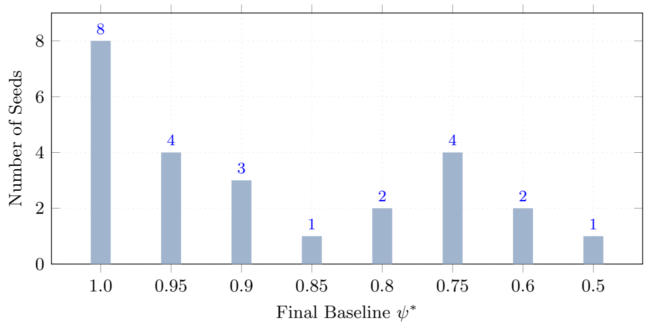


Figure 3: Minimal truncation $\psi^\star$ for fault reveling inputs in MNIST ($n = 25$).

Lowering ψ can also *salvage seeds* that fail the baseline at $\psi = 1.0$, moving them toward regions where they change the SUTs behavior and remain within the perceptual threshold. These salvaged seeds tend to lie closer to decision boundaries, increasing the likelihood of flips. Additionally, truncation reshapes the classifier's confidence landscape, sometimes surfacing rival classes that were suppressed at $\psi = 1.0$, a phenomenon we term *target revelation*. For example, Fig. 3 shows the distribution of $\psi^\star$ across human-validated, fault-revealing seeds in MNIST, one of the datasets of our study.

4 Empirical Study

4.1 Research Questions

We investigate two key questions:

RQ$_1$ (effectiveness) *How does truncation affect the validity, diversity, and fault detection of generated test inputs?* This question evaluates the core trade-off introduced by truncation. Lower truncation levels increase image realism and classifier confidence but may reduce diversity, potentially limiting fault discovery. Understanding this balance is important to selecting truncation settings that maximize both test validity and effectiveness.

RQ$_2$ (comparison) *How do the style-mixing and first-flip methods compare in terms of efficiency and fault detection?* This question contrasts two strategies for exploring the latent space: (i) *style-mixing*, which searches along semantic directions via layer-wise interpolation, and (ii) *first-flip*, which relies on truncation descent. Comparing their performance clarifies whether structured latent manipulations are necessary or if truncation alone suffices for generating meaningful, fault-revealing inputs.

4.2 Experimental Setup, Metrics, and Procedure

4.2.1 Objects of Study. We evaluate the proposed approach on three benchmark datasets: MNIST (28×28 grayscale) [19], Fashion-MNIST (28×28 grayscale) [33], and CIFAR-10 (32×32 color) [17]. For each dataset, we employ pretrained class-conditional StyleGAN2(-ADA) generators as test input producers from existing work [32]. The corresponding classifiers used are Torchvision baselines: a small CNN for MNIST and Fashion-MNIST, and a ResNet-18 for CIFAR-10 [8], also from existing work [32].

Truncation is explored over fixed budgets (1), and an adaptive schedule (2) is used to automatically locate the minimal truncation value $\psi^\star$ that produces valid candidates.

$$\psi_{\text{fixed}} = \{1.0,\ 0.9,\ 0.8,\ 0.7,\ 0.6,\ 0.5\} \tag{1}$$

$$\psi_{\text{adapt}} = \{1.0,\ 0.95,\ 0.90,\ 0.85,\ 0.80,\ 0.75,\ 0.70,\ 0.60,\ 0.50\} \tag{2}$$

Each generator is evaluated with 20 latent seeds per class for MNIST and Fashion-MNIST, and 30 seeds per class for CIFAR-10. Identical seed lists are reused across budgets to ensure comparability.

4.2.2 Evaluation Metrics. We assess the impact of truncation using four operational metrics that jointly capture the realism, diversity, and fault-revealing power of generated test inputs:

- **Baseline Acceptance.** For a seed z and class c, the baseline image $x_{1.0} = G(z, c; 1.0)$ is accepted if it meets three classifier-based criteria: (1) $\arg\max h(x_{1.0}) = c$ (predicted as the correct class); (2) top-class confidence $p_{(1)} \geq p_{\min}$; (3) confidence margin $p_{(1)} - p_{(2)} \geq \delta$.
- **Screening Validity.** For each truncation level b, the generated image $x_b = G(z, c; b)$ must pass both the classifier gate above and a threshold of SSIM = 0.95 and L2 = 0.2 that is fixed for every dataset. This step filters out visually implausible or off-mode samples before human inspection [32].
- **Human-Validated Validity.** Screened images are independently reviewed by human annotators to confirm whether they are class-plausible. This step captures true semantic validity that cannot be guaranteed by automated proxies.
- **Diversity and Fault Detection.** Diversity is quantified as the mean pairwise perceptual distance (LPIPS) among all human-validated samples for a given (c, b), measuring coverage of the visual input space. Fault detection is the proportion of validated samples that cause a misclassification, i.e., $\arg\max h(x) \neq c$, indicating the generator's ability to reveal model weaknesses [36].

4.2.3 Procedure. For each dataset, we generate test inputs under the fixed and adaptive truncation schedules and apply the two probing methods (style-mixing and first-flip). Each generated image undergoes automated screening and, if accepted, human verification. We then report (i) screening and validation rates per truncation level, (ii) diversity among validated inputs, (iii) fault detection ratios, and (iv) the relative efficiency of the two probing methods. Together, these measurements allow us to analyze how truncation affects the trade-off between *validity*, *diversity*, and *fault revelation*, and to assess whether adaptive truncation or specific probing strategies offer systematic advantages.

4.2.4 Human Validation. To obtain human validity, we conducted an annotation study with 2 PhD students in computer science, one of them is an author of this paper. For each dataset and class, we prepared a folder containing the generated flipped test cases and removed all metadata about their origin (e.g., value of ψ, generation technique, and classifier outputs). Annotators therefore only saw the raw images and the class. Annotator were instructed to answer a binary question: *"Is this image a valid example of this class?"* (yes/no). We mark an image as *human-valid* if the annotator answered "yes" for its class.

4.3 Effect of Truncation on Validity, Diversity, and Fault Detection (RQ$_1$)

Table 1 presents the results for all datasets and compared techniques. The results are averaged over the 25 runs. Regarding MNIST, without regularization (ψ=1.0), the human-validated rate is 48% (12/25) with a cost of 42.00 seeds per validated case. As ψ decreases, validation improves and effort drops, peaking at ψ=0.6 with 80% human-validated pairs and 22.75 seeds/valid. Pushing regularization further to ψ=0.5 slightly reduces validation (72%) and yields marginal efficiency gains compared to ψ=0.6.

The *adaptive* strategy achieves the best efficiency: it reaches 15 human-validated pairs using only 161 seeds in total (10.73 seeds/valid), i.e., less than half of the effort required at ψ=0.6. This confirms the practical value of per-seed regularization: instead of discarding borderline seeds at ψ=1.0, adaptively "salvaging" them near $\psi^\star$ yields more fault-prone tests at lower cost (see Fig. 5).

Although Table 1 primarily reports acceptance/efficiency, qualitative inspection (e.g., Fig. 5) shows that ψ≈0.6 maintains sufficient visual spread to reveal decision-boundary behavior. Extremely low ψ would risk collapsing diversity; our results suggest that mild to moderate regularization (and especially the adaptive strategy) strikes a better balance between human-validated realism and the variety needed to expose faults. Similar reasoning applies to the other datasets.

Table 1: Results across datasets, truncation levels, and probing techniques. Each entry reports the number of sampled seeds, *human-validated* test inputs, validation rate, mean pairwise perceptual diversity (LPIPS), fault detection rate, and efficiency (seeds per validated input).

Dataset	Technique	Setting	Seeds	Human-val	Human-Val. Rate	Diversity	Fault Rate
MNIST	Style-Mixing	No trunc.	504	11	44%	0.193914	2.18%
		ψ=0.9	518	12	48%	0.223992	2.31%
		ψ=0.8	461	16	64%	0.214914	3.47%
		ψ=0.7	549	14	56%	0.189029	2.55%
		ψ=0.6	455	**20**	**80%**	0.152080	4.39%
		ψ=0.5	448	17	68%	0.142870	3.79%
		Adaptive	**161**	15	60%	**0.252290**	**9.31%**
	Truncation-Only	Gradual trunc.	499	19	76%	0.155862	3.80%
Fashion-MNIST	Style-Mixing	No trunc.	431	23	92%	**0.239928**	5.33%
		ψ=0.9	502	24	96%	0.235572	4.78%
		ψ=0.8	718	**25**	**100%**	0.173270	3.48%
		ψ=0.7	706	**25**	**100%**	0.202881	3.54%
		ψ=0.6	392	24	96%	0.165565	6.12%
		ψ=0.5	526	**25**	**100%**	0.170068	4.75%
		Adaptive	**169**	22	88%	0.213573	**13.0%**
	Truncation-only	Gradual trunc.	93	25	100%	0.227403	26.8%
CIFAR-10	Style-Mixing	No trunc.	275	13	52%	**0.423589**	4.72%
		ψ=0.9	249	13	52%	0.404477	5.22%
		ψ=0.8	252	16	64%	0.400013	6.34%
		ψ=0.7	249	15	60%	0.406339	6.02%
		ψ=0.6	158	**17**	**68%**	0.382690	10.7%
		ψ=0.5	**118**	16	64%	0.338466	**13.5%**
		Adaptive	255	13	52%	0.421139	5.09%
	Truncation-only	Gradual trunc.	266	16	64%	0.386001	6.01%

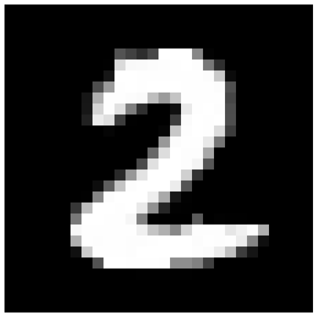
RQ$_1$ (effectiveness). Our experiment revealed that mild–moderate latent regularization ($\psi \approx 0.6$) maximizes human-validated frontier yield on MNIST. The adaptive policy minimizes human effort (seeds/valid) by salvaging hard seeds. Qualitative analysis shows that diversity remains adequate at these settings to expose faults, whereas overly strong regularization offers lowering returns.

4.4 Style-Mixing vs. Truncation-Only First-Flip (RQ$_2$)

The two strategies appear to be complementary. *First-flip* is lightweight and effective for quickly revealing faults under a truncation schedule (see Fig. 6); *style-mixing* provides semantically guided boundary traversal with finer control, at the cost of extra search (see Fig. 5).

Across the same truncation settings used for RQ$_1$, first-flip requires fewer design choices (no rival selection or layer sets) and rapidly identifies misbehaviours by descending ψ. This makes it a strong configuration when computing time is limited. Style-mixing, in contrast, steers the latent representation toward specific rival classes via layer-wise interpolation and binary search; this extra complexity typically yields cleaner, semantically interpretable boundary crossings and can improve human acceptance for borderline cases, but incurs additional tuning (layers L, step budget T, rival selection).

On MNIST, runs that reached the fixed quota of 25 frontiers showed that first-flip benefited markedly from adaptive seed salvage (lower seeds/valid) and produced a higher fraction of validated faults in our MNIST experiments, while style-mixing produced boundary examples with more controllable semantics. In short, if the goal is *fast fault surfacing*, first-flip under adaptive ψ is preferable; if the goal includes *interpretable semantics* for analysis and debugging, style-mixing provides that control at moderate extra cost. Similar reasoning applies to the other datasets.


(a) Baseline (ψ=1.00): human-valid image, source class *2* with $p \approx 1.00$; others $\approx$ 0.

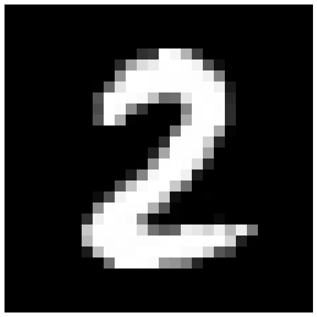

(b) Truncated (ψ=0.75): flips to *9* with p=1.00 using *truncation only* (untargeted).

Figure 4: Target revelation via truncation. Lowering ψ reshapes the rival landscape; this strongly peaked, human-valid seed flips under truncation alone.

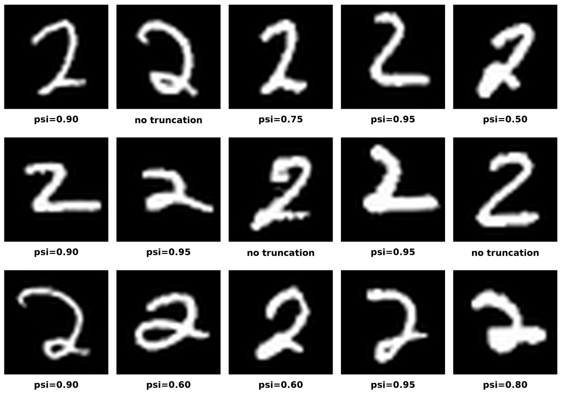


Figure 5: MNIST: adaptive truncation, human-validated, fault-revealing test cases. Images are annotated with respective truncation ψ levels.

RQ$_2$ (comparison). First-flip is the most efficient for fault discovery under latent regularization; style-mixing complements it by adding semantic control over boundary direction. Using first-flip to triage seeds (possibly with adaptive ψ) and style-mixing for deeper analysis provides strong overall coverage of boundary behaviors.

4.5 Threats to Validity

4.5.1 Internal validity. Internal validity concerns whether the observed effects truly result from the factors studied, rather than from uncontrolled variables. A first possible threat is the selection of latent seeds, which could influence the results if some seeds inherently produce more realistic or diverse samples than others. This risk was mitigated by reusing the same seed lists across all truncation budgets and probing methods. Another issue is that very strong truncation may cause mode collapse, reducing diversity and artificially inflating validity. To avoid this, we limited truncation

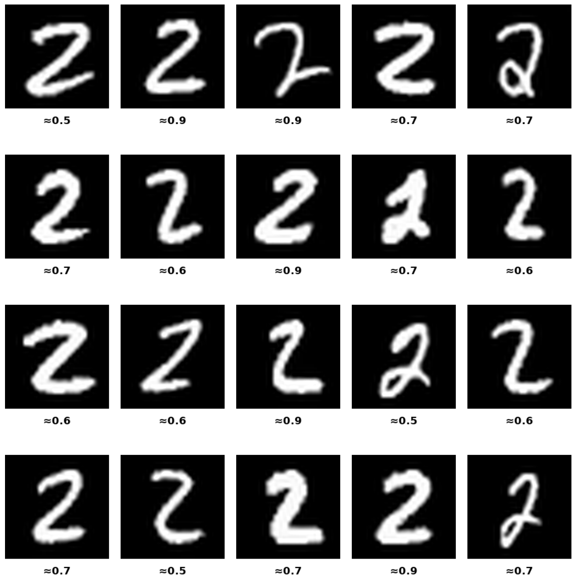


Figure 6: First-flip examples. Cases from MNIST where the first flip occurs at $\psi^\star$.

sweeps to values of $\psi \geq 0.5$. Finally, the degree of latent disentanglement varies across datasets and classes, which can make the effects of truncation or layer-wise edits less consistent. This limitation reflects an intrinsic property of current generative models.

4.5.2 Construct validity. Construct validity relates to how accurately our measurements reflect the intended concepts of validity, diversity, and fault detection. Human annotation introduces subjectivity, since different annotators may disagree on what constitutes a class-plausible image. We reduced this risk by providing clear annotation guidelines and, where possible, using multiple annotators. Similarly, the perceptual similarity metric (LPIPS) used to filter samples is only an approximation of true semantic similarity. While imperfect, it provides a consistent and reproducible basis for comparing truncation levels and probing strategies across datasets.

4.5.3 External validity. External validity concerns the generalizability of our findings. Our experiments were conducted on small-scale image datasets with relatively simple classifiers. The observed relationships between truncation, validity, and diversity may differ in larger or more complex domains. Moreover, the conclusions apply specifically to StyleGAN-based generators, and the behavior of truncation may vary in other architectures, such as diffusion or transformer-based generative models. Extending this analysis to additional data modalities and architectures is therefore an important direction for future work.

To ensure transparency and replicability, we release all experimental scripts (including seed lists, fixed-noise configurations), JSON logs, and plotting templates to reproduce every figure and table in an anonymous repository (https://github.com/deeptest26-anon/Latent-Regularization-in-Generative-Test-Input-Generation).

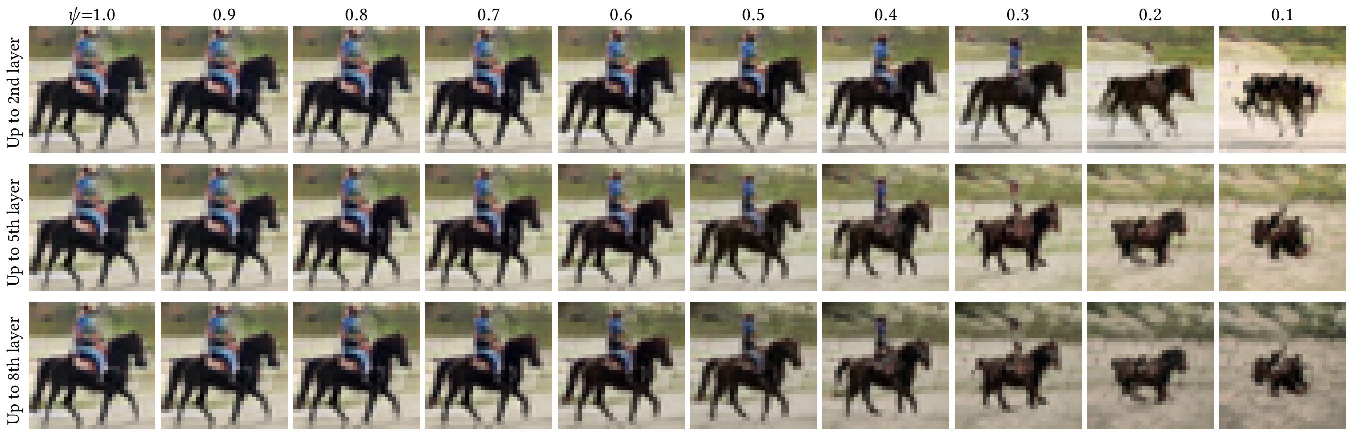
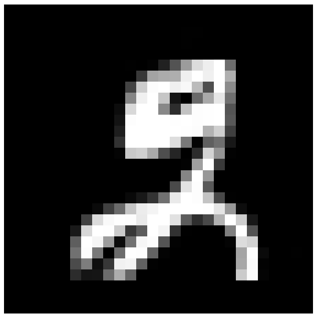
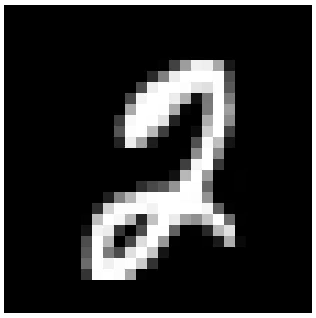


Figure 7: Layer-specific truncation sweeps (one seed). Columns show decreasing ψ from 1.0 to 0.1; rows indicate which layers are truncated: up to layer 2 (coarse), up to layer 5 (coarse+middle), and up to layer 8 (all layers).



(a) **Baseline (ψ=1.00): human-invalid but classifier-correct for the source class.**



(b) **Truncated (ψ=0.55): human-valid and also flips to a rival class (truncation-only).**

Figure 8: Truncation-only search. Flipping with refinement example. Here, truncation refines a human-invalid baseline into a human-valid image and helps flip the classifier's prediction by adding truncation (ψ=0.55) on all layers.

5 Discussion

Our results show that truncation can serve as a simple yet effective control for enhancing the quality of generated test inputs in latent-space testing frameworks such as Mimicry. Moderate truncation levels ($\psi \approx 0.6$) tend to maximize the fraction of human-validated valid and fault-revealing test cases, striking a balance between semantic realism and diversity.

The adaptive truncation mechanism further improves efficiency by salvaging seeds that would otherwise be discarded, leading to a richer and more fault-prone test pool. This adaptivity makes truncation-based refinement particularly suitable when annotation or computational budgets are limited.

Between the two probing methods, *style-mixing* provides structured, semantically meaningful perturbations but requires additional computation and hyperparameter tuning. In contrast, the *truncation-only first-flip* approach offers a lightweight, diagnostic alternative that can reveal classifier weaknesses with minimal setup. Combining both techniques allows a comprehensive exploration of decision boundaries, from coarse latent refinements to fine-grained semantic traversals.

6 Conclusion and Future Work

This study investigated how truncation—a common latent-space regularization technique—affects the generation of test inputs for deep learning classifiers. We quantified its impact on human-verified validity, diversity, and fault detection, and introduced a simple yet effective *first-flip* probe based solely on truncation. We further compared this approach with a truncation-assisted *style-mixing* probe that explores the latent space through semantic interpolation.

Our findings show that moderate truncation levels and adaptive seed refinement yield the best balance between validity, diversity, and fault revelation. These results provide practical guidance and actionable defaults for configuring generative test input generators, as well as a reproducible protocol to support future research in deep learning testing.

Future extensions of this study could explore truncation effects across StyleGAN's layers (see Fig. 7), as well as related generator-level controls such as synthesis noise or layer-selection choices, and in more complex generative architectures, such as diffusion or transformer-based models, where latent representations and fidelity controls differ fundamentally from GANs. Finally, integrating automated validity estimation and adaptive truncation scheduling could reduce reliance on human annotation, advancing scalable and self-adaptive test generation for AI-based systems.

References

[1] David Bau, Jun-Yan Zhu, Hendrik Strobelt, Bolei Zhou, Joshua B. Tenenbaum, William T. Freeman, and Antonio Torralba. 2018. GAN Dissection: Visualizing and Understanding Generative Adversarial Networks. *arXiv preprint arXiv:1811.10597* (2018).

[2] Wieland Brendel, Jonas Rauber, and Matthias Bethge. 2018. Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models. In *Proc. ICLR*.

[3] Andrew Brock, Jeff Donahue, and Karen Simonyan. 2019. Large Scale GAN Training for High Fidelity Natural Image Synthesis. In *Proc. ICLR*.

[4] Jianbo Chen, Michael I. Jordan, and Martin J. Wainwright. 2020. HopSkipJumpAttack: A Query-Efficient Decision-Based Attack. In *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*.

[5] Swaroopa Dola, Rory McDaniel, Matthew B Dwyer, and Mary Lou Soffa. 2024. Cit4dnn: Generating diverse and rare inputs for neural networks using latent space combinatorial testing. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*. 1–13.

[6] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Proc. NeurIPS*.

[7] Jianmin Guo, Yao Shen, Yue Huang, et al. 2019. DLFuzz: Differential Fuzzing Testing of Deep Learning Systems. In *Proc. ICSE*.

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *Proc. CVPR*.

[9] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. 2017. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. In *Proc. NeurIPS*.

[10] Erik Härkönen, Aaron Hertzmann, Jaakko Lehtinen, and Sylvain Paris. 2020. GANSpace: Discovering Interpretable GAN Controls. In *Proc. NeurIPS*.

[11] Sungmin Kang, Robert Feldt, and Shin Yoo. 2020. SINVAD: Search-based Image Space Navigation for DNN Image Classifier Test Input Generation. In *Proc. ISSTA*. ACM, 521–528. doi:10.1145/3387940.3391456

[12] Tero Karras, Miika Aittala, Janne Hellsten, et al. 2021. Alias-Free Generative Adversarial Networks, In Proc. NeurIPS. *NeurIPS*.

[13] Tero Karras, Miika Aittala, Janne Hellsten, Samuli Laine, Jaakko Lehtinen, and Timo Aila. 2020. Training Generative Adversarial Networks with Limited Data. In *Proc. NeurIPS*.

[14] Tero Karras, Samuli Laine, and Timo Aila. 2019. A Style-Based Generator Architecture for Generative Adversarial Networks. In *Proc. CVPR*.

[15] Tero Karras, Samuli Laine, Miika Aittala, et al. 2020. Analyzing and Improving the Image Quality of StyleGAN. In *Proc. CVPR*. 8110–8119.

[16] Jinhan Kim, Robert Feldt, and Shin Yoo. 2019. Guiding Deep Learning System Testing using Surprise Adequacy. In *Proc. ICSE*.

[17] Alex Krizhevsky. 2009. *Learning Multiple Layers of Features from Tiny Images*. Technical Report. University of Toronto.

[18] Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. 2019. Improved Precision and Recall Metric for Assessing Generative Models. In *Proc. NeurIPS*.

[19] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-Based Learning Applied to Document Recognition. *Proc. IEEE* 86, 11 (1998), 2278–2324.

[20] Lei Ma, Felix Juefei-Xu, Fuyuan Zhang, et al. 2018. DeepGauge: Multi-Granularity Testing Criteria for Deep Learning Systems. In *Proc. ASE* (Montpellier, France) *(ASE 2018)*. ACM, New York, NY, USA, 120–131. doi:10.1145/3238147.3238202

[21] Maryam Maryam, Matteo Biagiola, Andrea Stocco, and Vincenzo Riccio. 2025. Benchmarking Generative AI Models for Deep Learning Test Input Generation. In *2025 IEEE Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 174–185.

[22] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. 2016. DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. In *Proc. CVPR*.

[23] Nusrat Jahan Mozumder, Felipe Toledo, Swaroopa Dola, and Matthew B Dwyer. 2025. RBT4DNN: Requirements-based Testing of Neural Networks. *arXiv preprint arXiv:2504.02737* (2025).

[24] Augustus Odena, Catherine Olsson, David G. Andersen, and Ian Goodfellow. 2018. TensorFuzz: Debugging Neural Networks with Coverage-Guided Fuzzing. *arXiv preprint arXiv:1807.10875* (2018).

[25] Kexin Pei, Yinzhi Cao, Junfeng Yang, and Suman Jana. 2017. DeepXplore: Automated Whitebox Testing of Deep Learning Systems, In Proc. SOSP (Shanghai, China). *Commun. ACM* 62, 11, 1–18. doi:10.1145/3132747.3132785

[26] Vincenzo Riccio and Paolo Tonella. 2020. Model-Based Exploration of the Frontier of Behaviours for Deep Learning System Testing. In *Proc. ESEC/FSE*. ACM, 876–888. doi:10.1145/3368089.3409730

[27] Mehdi S. M. Sajjadi, Olivier Bachem, and Mario Lucic. 2018. Assessing Generative Models via Precision and Recall. In *Proc. NeurIPS*.

[28] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. 2016. Improved Techniques for Training GANs. In *Proc. NeurIPS*.

[29] Axel Sauer, Katja Schwarz, and Andreas Geiger. 2022. StyleGAN-XL: Scaling StyleGAN to Large Datasets. In *Proc. SIGGRAPH*. 1–10.

[30] Yujun Shen and Bolei Zhou. 2021. Closed-Form Factorization of Latent Semantics in GANs. In *Proc. CVPR*.

[31] Yuchi Tian, Kexin Pei, Suman Jana, and Baishakhi Ray. 2018. DeepTest: Automated Testing of Deep-Neural-Network-Driven Autonomous Cars. In *Proc. ICSE*.

[32] Oliver Weißl, Amr Abdellatif, Xingcheng Chen, Giorgi Merabishvili, Vincenzo Riccio, Severin Kacianka, and Andrea Stocco. 2025. Targeted Deep Learning System Boundary Testing. *ACM Trans. Softw. Eng. Methodol.* (Oct. 2025). doi:10.1145/3771557

[33] Han Xiao, Kashif Rasul, and Roland Vollgraf. 2017. Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv preprint arXiv:1708.07747* (2017).

[34] Xiaofei Xie, Lei Ma, Felix Juefei-Xu, et al. 2019. DeepHunter: A Coverage-Guided Fuzz Testing Framework for Deep Neural Networks. In *Proc. ISSTA*.

[35] Mengshi Zhang, Yuqun Zhang, Lingming Zhang, Cong Liu, and Sarfraz Khurshid. 2018. DeepRoad: GAN-Based Metamorphic Testing and Input Validation Framework for Autonomous Driving Systems. In *Proc. ASE*. ACM. doi:10.1145/3238147.3238187

[36] Richard Zhang, Phillip Isola, Alexei A. Efros, et al. 2018. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *Proc. CVPR*.

[37] Tahereh Zohdinasab, Vincenzo Riccio, Alessio Gambi, and Paolo Tonella. 2021. DeepHyperion: Exploring the Feature Space of Deep Learning-Based Systems through Illumination Search. In *Proc. ISSTA (ISSTA '21)*. ACM, 79–90. doi:10.1145/3460319.3464811

[38] Tahereh Zohdinasab, Vincenzo Riccio, Alessio Gambi, and Paolo Tonella. 2023. Efficient and Effective Feature Space Exploration for Testing Deep Learning Systems. *ACM Transactions on Software Engineering and Methodology* 32, 2 (2023), 49:1–49:38. doi:10.1145/3544792